ANL-7109

ANL-7109

Argonne National Laboratory

A CHECKER-PLAYING PROGRAM FOR THE IPL-VC COMPUTER

by

W. R. Cowell and M. C. Reed

Printed in USA. Price $1.00. Available from the Clearinghouse for Federal Scientific and Technical Information, National Bureau of Standards, U. S. Department of Commerce, Springfield, Virginia

ANL-7109
Mathematics and Computers
(TID-4500, 47th Ed.)
AEC Research and
Development Report

ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60440

A CHECKER-PLAYING PROGRAM FOR THE IPL-VC COMPUTER

by

W. R. Cowell and M. C. Reed

Applied Mathematics Division

October 1965

Operated by The University of Chicago
under
Contract W-31-109-eng-38
with the
U. S. Atomic Energy Commission

TABLE OF CONTENTS

A CHECKER-PLAYING PROGRAM FOR THE IPL-VC COMPUTER

by

W. R. Cowell and M. C. Reed

INTRODUCTION

Our purpose was to write a computer program to perform a task that could be considered a complex, intellectual activity when performed by humans. The program was to be sufficiently flexible so that substantial modifications to improve its performance would be easy and natural as dictated by experience with its usage.

The game of checkers was selected as an appropriate task for the same reasons given by Samuel[1] and for the additional important reason that Samuel's very successful checker player already existed, making possible comparisons in terms of programming effort, running time, and effectiveness of the program.

Our approach to the design of a checker player differed from Samuel's in two basic respects:

1) We wrote the program in a high-level, list-processing language (IPL-V) embodying the flexibility described above.

2) Our program "learned" by humanly inspired modifications rather than by automatic adjustment of parameters.

About three man-months (by programmers with no previous experience in IPL-V) were required to produce the first version, a player with a very elementary strategy.* But the price paid for this programming ease was high. A single board position is stored in 66 words, by comparison with four words in Samuel's program. Again, Samuel's program for the IBM-704 requires 2.6 msec to determine the legal moves from a given board position. Our program on the 704 IPL-V interpreter requires 2-3 sec for the same determination--a discouraging three orders of magnitude slower. It was clear that the cost of machine time would not permit serious experimentation with play against human opponents.

---

*Using the language of Sec. (2) it used the principle routine PCR with a maximum cumulative node count of 1.

A CHECKER-PLAYING PROGRAM FOR THE IPL-VC COMPUTER

by

W. R. Cowell and M. C. Reed

INTRODUCTION

Our purpose was to write a computer program to perform a task that could be considered a complex, intellectual activity when performed by humans. The program was to be sufficiently flexible so that substantial modifications to improve its performance would be easy and natural as dictated by experience with its usage.

The game of checkers was selected as an appropriate task for the same reasons given by Samuel[1] and for the additional important reason that Samuel's very successful checker player already existed, making possible comparisons in terms of programming effort, running time, and effectiveness of the program.

Our approach to the design of a checker player differed from Samuel's in two basic respects:

1) We wrote the program in a high-level, list-processing language (IPL-V) embodying the flexibility described above.

2) Our program "learned" by humanly inspired modifications rather than by automatic adjustment of parameters.

About three man-months (by programmers with no previous experience in IPL-V) were required to produce the first version, a player with a very elementary strategy*. But the price paid for this programming ease was high. A single board position is stored in 66 words, by comparison with four words in Samuel's program. Again, Samuel's program for the IBM-704 requires 2.6 msec to determine the legal moves from a given board position. Our program on the 704 IPL-V interpreter requires 2-3 sec for the same determination--a discouraging three orders of magnitude slower. It was clear that the cost of machine time would not permit serious experimentation with play against human opponents.

---

*Using the language of Sec. (2) it used the principle routine PCR with a maximum cumulative node count of 1.

A significant improvement should result from the development of an interpreter for a larger and faster machine (the CDC-3600) available to us. However, when Donald Hodges extended the 3600 system to include a hardware list processor,[2] we transferred our activity to Engine No. 2, a computer with the IPL-V instruction set.

Our program ran 37 times faster on IPL-VC than on the IBM-704 interpretive system. Study of the program indicated that if the IPL-VC system had used only one 32,768-word bank of Control Data 3600 memory instead of two, then this speed advantage would probably have been doubled.

THE CHECKER-PLAYING PROGRAM IN IPL-V

Since processes are available in the IPL-V language for dealing with the special format of description-lists,[3] this format was used for storing the rules for checker play. Thus, for example, a list associated with black checker moves has a description list in which symbols designating the 32 squares are "attributes" whose "values" are names of lists of squares to which a black checker may legally move from the attribute square provided the target square is empty and no jumps are available to black. Similarly, description lists are used to store potential legal moves for white checkers and for kings. Potential one-checker jumps for the various types of pieces are stored, and jumps involving more than one checker are constructed recursively during the determination of the legal moves.

In a similar manner, a description list is used to represent a given board position. The attributes are squares, and the value of each square is a symbol designating its contents.

Fundamental to machine play of a board game such as checkers is the construction of a "look-ahead" tree of future positions. The nodes of the tree represent the positions and the branches of possible moves. At any move in the game we consider a tree whose origin node corresponds to the present board position. To store the tree in memory, we need store only the present board position, together with a move associated with each node. Then each node corresponds to the position obtainable by performing the unique sequence of moves from the origin node.

In the program, a node is represented by a list of squares defining a move. The list contains the square moved from, the square moved to, and the squares jumped over, if any. This node list also contains the names of data terms which serve as valuations for the corresponding board position in a manner to be described. The node lists are describable, and the description lists provide branching information. The attributes on the

A significant improvement should result from the development of an interpreter for a larger and faster machine (the CDC-3600) available to us. However, when Donald Hodges extended the 3600 system to include a hardware list processor,[2] we transferred our activity to Engine No. 2, a computer with the IPL-V instruction set.

Our program ran 37 times faster on IPL-VC than on the IBM-704 interpretive system. Study of the program indicated that if the IPL-VC system had used only one 32,768-word bank of Control Data 3600 memory instead of two, then this speed advantage would probably have been doubled.

THE CHECKER-PLAYING PROGRAM IN IPL-V

Since processes are available in the IPL-V language for dealing with the special format of description-lists,[3] this format was used for storing the rules for checker play. Thus, for example, a list associated with black checker moves has a description list in which symbols designating the 32 squares are "attributes" whose "values" are names of lists of squares to which a black checker may legally move from the attribute square provided the target square is empty and no jumps are available to black. Similarly, description lists are used to store potential legal moves for white checkers and for kings. Potential one-checker jumps for the various types of pieces are stored, and jumps involving more than one checker are constructed recursively during the determination of the legal moves.

In a similar manner, a description list is used to represent a given board position. The attributes are squares, and the value of each square is a symbol designating its contents.

Fundamental to machine play of a board game such as checkers is the construction of a "look-ahead" tree of future positions. The nodes of the tree represent the positions and the branches of possible moves. At any move in the game we consider a tree whose origin node corresponds to the present board position. To store the tree in memory, we need store only the present board position, together with a move associated with each node. Then each node corresponds to the position obtainable by performing the unique sequence of moves from the origin node.

In the program, a node is represented by a list of squares defining a move. The list contains the square moved from, the square moved to, and the squares jumped over, if any. This node list also contains the names of data terms which serve as valuations for the corresponding board position in a manner to be described. The node lists are describable, and the description lists provide branching information. The attributes on the

description lists are indexing symbols (indicating branch 1, branch 2, etc.) and the value of each is the name of the node list corresponding to the node on the associated branch.

The depth of look-ahead is determined by a variable called the "cumulative node count." From a given node, we trace a unique path back to the origin node and determine the number of branches from each of the nodes on the path except the origin node itself. The number of such branches is the cumulative node count and serves as a gross measure of the complexity of the game tree. No further growth occurs from a given node when

1) The position is dead (the active player has no forced jumps), and either
2) The cumulative node count exceeds some assigned maximum, or
3) The relative piece strength is different from that at the origin (king = 1.5 checkers).

The power of list processing as a means for dynamically assigning storage is evident in those parts of the program in which the storage requirements are apparent only at run time. The look-ahead tree is a prime example. IPL-V provided the capability for constructing and manipulating this tree by means of recursive routines. On the other hand the use of lists for representing board positions, storing checker rules, and dealing with other fixed storage requirements was a concession to the nature of the programming language and was generally very demanding of space and of processing time.

To describe the strategy for play, we shall define a principle routine as a program that accepts as input a given board position P and a nonempty subset S of the set of legal moves from that position by the active player, and that produces as output a nonempty subset of S. The output moves are those among the input moves that qualify as being best (and equally acceptable) with respect to some principle of play.

Each principle routine uses the look-ahead tree of potential play. To each position at which growth is terminated, an evaluation or score is assigned. This score measures the merit of the future position relative to the given principle to the player who is active at P. Minimaxing[1] through the tree to the position P will enable selection of those moves that will result (so far as the scope of the tree permits examination) in equivalent scores. Only scores relative to the same principle are compared. Thus the numerical value of the score has no intrinsic meaning but only serves to establish the rank of a position.

A list of principle routines provides a technique for selecting a move. The legal moves from a given position are subjected to the first routine in the list, and each output set of moves serves as input to the next routine (which uses an appropriately pruned tree). If only one move is input, it is assumed to qualify relative to any principle. If the final principle routine has more than one output, a random selection is made. The principle routines used by the program are described in Appendix A.

The strategy applied by the program at a given move is determined when

1) A list of principle routines is selected, and

2) The maximum cumulative node count is assigned.

The program may select different lists of principle routines in different parts of the game, and the maximum cumulative node count may reflect the number of pieces on the board. Between games, the principle routines and the "overseer" routine that selects the strategy may be modified as need is discovered through experience with the program. IPL-V proved to be an excellent vehicle for constructing a program with the required degree of flexibility.

When the program that had run on the 704 was running successfully on the IPL-VC system, a series of games was played between the program and various human opponents. The human players were amateurs of modest checker-playing attainments. The 3600 console typewriter was used for input-output. The games were leisurely replayed later by the opponents and the authors, and this experience with the program dictated the construction of additional principle routines and their assembly into lists defining strategy. Examination of the machine's performance was aided by building into the program a means of analyzing the human opponent's play and comparing it with the play of the machine. A dual set of principle routines was used for this purpose, and when a human move would have been rejected by some principle routine, this fact was noted for later consideration.

Appendix B contains four games played by the program against different opponents. The comments are those of Mr. Leo Levitt of Los Angeles, California, a ranking checker master who has held several state and national championships. When these games were sent to Mr. Levitt for comment, he knew only that at least one of the players was a computer program. In Games I and II, the principle routine list was PCR, KRP, KD, CRAMP, MOBIL, and CENT, and the maximum cumulative node count was set at 1, 4, or 9 depending on the number of pieces on the board. In Game III, the maximum cumulative node count was set at 4 or 9 and the principle routine list was PCR, KRP, KD, and CENT, or the above list, depending on the number of pieces on the board. In Game IV, a somewhat more elaborate "overseer" was used, account being taken of whether play was in opening, middle, or end game and whether the machine was ahead or

A list of principle routines provides a technique for selecting a move. The legal moves from a given position are subjected to the first routine in the list, and each output set of moves serves as input to the next routine (which uses an appropriately pruned tree). If only one move is input, it is assumed to quality relative to any principle. If the final principle routine has more than one output, a random selection is made. The principle routines used by the program are described in Appendix A.

The strategy applied by the program at a given move is determined when

1) A list of principle routines is selected, and

2) The maximum cumulative node count is assigned.

The program may select different lists of principle routines in different parts of the game, and the maximum cumulative node count may reflect the number of pieces on the board. Between games, the principle routines and the "overseer" routine that selects the strategy may be modified as need is discovered through experience with the program. IPL-V proved to be an excellent vehicle for constructing a program with the required degree of flexibility.

When the program that had run on the 704 was running successfully on the IPL-VC system, a series of games was played between the program and various human opponents. The human players were amateurs of modest checker-playing attainments. The 1620 console typewriter was used for input-output. The games were leisurely replayed later by the opponents and the authors, and this experience with the program dictated the construction of additional principle routines and their assembly into lists defining strategy. Examination of the machine's performance was aided by building into the program a means of analyzing the human opponent's play and comparing it with the play of the machine. A dual set of principle routines was used for this purpose, and when a human move would have been rejected by some principle routine, this fact was noted for later consideration.

Appendix B contains four games played by the program against different opponents. The comments are those of Mr. Leo Levitt of Los Angeles, California, a ranking checker master who has held several state and national championships. When these games were sent to Mr. Levitt for comment, he knew only that at least one of the players was a computer program. In Games I and II, the principle routine list was PCR, KRP, KD, CRAMP, MOBIL, and CENT, and the maximum cumulative node count was set at 1, 4, or 9 depending on the number of pieces on the board. In Game III, the maximum cumulative node count was set at 4 or 9 and the principle routine list was PCR, KRP, KD, and CENT, or the above list, depending on the number of pieces on the board. In Game IV, a somewhat more elaborate "overseer" was used, account being taken of whether play

behind in piece strength. The results were disappointingly poorer than those with the very simple overseers used in the other games. Actually, this game between our program and this early version of Samuel's program was one of the poorest played. It ended in a meaningless "draw" with each player aimlessly moving a king back and forth.

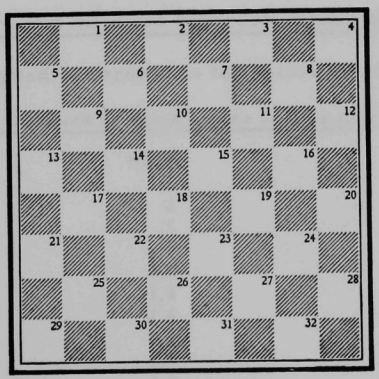
These games showed that we are able to construct a checker-playing program that was capable of reasonably good amateur play, though its play was erratic in quality. It beat one of its authors (Game III) in a creditable manner even though he was thoroughly aware of the details of its construction. It is of interest to note that Mr. Levitt was wrong in assessing which player in Games II and III was the machine.*

The flexibility of the program would permit the addition of further principle routines to deal with many special checker situations. Whether the program could be brought to the standard of master play through such additions is not certain. Playing time and memory space are being taxed at the present level of performance, and it seems likely that substantial improvements would depend on expanded facilities as much as on clever additions to the strategy.

REMARK

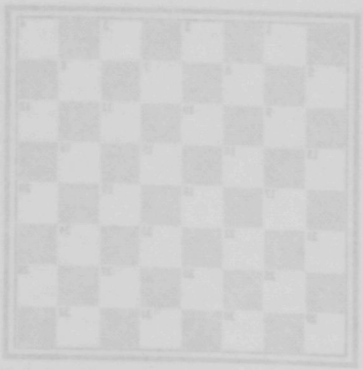
In the appendixes, the usual checker conventions are used.

1) The board is numbered as shown. In the starting position, black occupies the squares from 1 to 12, and white the squares from 21 to 32.



2) Black moves first. In the game listings, white moves are shown indented.

*Since Mr. Levitt did not know the caliber of the human players, he was at a disadvantage. In the absence of a deep understanding of the game, novices tend to follow a set of rather inflexible rules in their play.



APPENDIX A

Principle Routines Used by Program

For each principle routine, we give the method of assigning a score to the terminal node when black is the active player at the origin node.

PCR (Piece-count ratio)

$$S = \frac{\text{number of black checkers} + 1.5 \times \text{number of black kings}}{\text{number of white checkers} + 1.5 \times \text{number of white kings}}.$$

CENT (Control of the center)

S = total weight calculated from the following table less the dual weight calculated for white.

Square	Weight When Occupied by Black Checker	Weight When Occupied by Black King
9	1	1
10	2	2
11	1	1
14	4	8
15	3	6
18	2	4
19	4	8
22	0	1
23	0	2
24	0	1

KRP (Effective and efficient king-row protection)

S is assigned according to the following table.

Squares Occupied by Black Pieces on King Row	Score
None	0
1	3
2	3
3	3
4	1
1,2	5
1,3	9
1,4	5
2,3	5
2,4	5
3,4	2
1,2,3	8
1,2,4	7
1,3,4	8
2,3,4	4
1,2,3,4	6

KD (King development)

S = -n, where n is the number of black kings on the row of squares 29, 30, 31, and 32.

MOBIL (Mobility)

S = number of plausible moves available to black if black were active at given node. (A plausible move is a legal move that does not result in a jump sequence leading to a dead position for which the value of PCR is less than that at the given node.)

CRAMP (Cramping opponent)

S = -n, where n is the number of plausible moves (see above) available to white if white were active at the given node.

KD (King development)

S = -n, where n is the number of black kings on the row of squares 29, 30, 31, and 32.

MOBIL (Mobility)

S = number of plausible moves available to black if black were active at given node. (A plausible move is a legal move that does not result in a jump sequence leading to a dead position for which the value of PCH is less than that at the given node.)

CRAMP (Cramping opponent)

S = -n, where n is the number of plausible moves (see above) available to white if white were active at the given node.

APPENDIX B

Games Played

GAME I PROGRAM (BLACK) VS J. ALAN ROBINSON (WHITE)

10-14

22-18

11-15

18-11

8-15

24-20

4-8

23-19 Weak; 28-24 More natural

15-24

28-19

7-11 Odd

27-23; 25-22 Much better

14-17 Correct

21-14

9-27

32-23

11-15 Correct

19-10

6-15

23-18 Good

15-22

25-18

2-7

26-23

7-10

23-19

8-11

30-25 Looks like a machine move; 29-25 is much better and is more natural

5-9

25-21

10-14 A

19-15

14-23

15-8

12-16

20-11

3-12

11-7

9-14 Poor; 12-16 followed by 16-20 and 23-27 draws

7-2

12-16

2-7

16-19

7-10

23-26

31-22

19-24

10-17

24-28

22-18

28-32

18-15

32-28

15-10

Black resigns

A 1-5,29-25,9-14,18-9,5-14,31-27,11-15,27-24,15-18,20-16,10-15, Black wins.

↓

IF 31-27,10-14,27-23,9-13,18-9,5-14,29-25,13-17,23-18,14-23, 21-14, 23-26,25-22,26-30,22-18,30-26,19-15,3-8 and Black wins against 15-10 with 26-22 and against 14-9 or 14-10 with 26-23.

GAME II PROGRAM (BLACK) VS JOHN REYNOLDS (WHITE)

10-14

23-18

14-23

27-18

9-14

18-9

5-14

22-18 Very weak; 26-23 is a natural here

14-23

26-19

11-15 Bad; 11-16 should be good enough to win

19-10

7-14

25-22

2-7 Meaningless; 8-11 or 12-16 better

22-17

12-16

17-10

7-14

31-26

8-11

26-23

4-8; 16-20 Much better

24-20 Correct

6-10

29-25

10-15

28-24

15-18

32-27

8-12

24-19

1-6

19-15

6-9 Excellent!

15-8

16-19

23-16

12-19

25-22

18-25

8-4

25-29

20-16

29-25

16-12

14-18

4-8

25-22

8-11; 8-4 Has a better chance

18-23

27-18

22-8

21-17

19-23

17-13

9-14

White resigns

Black's play seemed almost human! Am I right?

GAME III PROGRAM (BLACK) VS WAYNE COWELL (WHITE)

9-14
23-19
11-15
22-17
5-9
26-23
7-11
23-18 Mistake;
17-13 better
14-23
27-18
15-22
25-18
9-14
18-9
6-22
30-25
11-16
25-18
16-23
24-19
2-7
21-17
7-11
18-14
11-15
14-7
15-24
28-19
3-10
32-28
8-11
28-24
23-27
24-20
27-32
17-13
11-15
20-16
15-24
16-11
32-28
11-7
24-27
31-24
28-19
7-2
10-14
13-9
19-24
29-25
24-19
2-7
19-15
25-22
15-11
7-16
12-19
9-5
19-24
22-17
14-21
White loses

Black played reasonably well here, winning methodically after the mistake.

GAME IV PROGRAM (BLACK) VS 704 VERSION OF SAMUEL'S PROGRAM (WHITE)

9-14	10-6	23-27
22-17	8-12	6-2
10-15	15-8	27-31
17-10	4-11	2-7
7-14	6-1	31-26
25-22	5-9	24-19
2-7	26-23	26-31
24-19	18-27	20-16
15-24	31-24	31-27
27-20	3-8	7-10
6-10	1-5	27-32
22-17	16-19	1-5
5-9	24-15	32-27
28-24	11-18	5-9
11-15	25-22	27-32
23-19	18-25	10-7
7-11	29-22	32-27
32-28	8-11	7-10
15-18	22-17	27-32
19-15	9-13	10-7
10-19	17-10	32-27
24-15	11-15	7-10
12-16	10-6	Draw?
17-10	15-18	
9-14	28-24	
30-25	18-23	
1-5	5-1	

White wins with anything. Apparently the machine can't get a king out of a corner.

GAME IV PROGRAM (BLACK) VS 704 VERSION OF SAMUEL'S PROGRAM (WHITE)

White wins with anything. Apparently the machine can't get a king out of a corner.

ACKNOWLEDGMENTS

We wish to thank the human opponents who played checkers with the program, and Mr. Leo Levitt, who furnished expert commentary on the game of checkers and on several of the games played by the program.

REFERENCES

1. Samuel, A. L., Some Studies in Machine Learning Using the Game of Checkers, IBM Journal of Research and Development 3, No. 3 (July 1959).

2. Hodges, Donald, IPL-VC, A Computer System Having the IPL-V Instruction Set, Argonne National Laboratory, ANL-6888 (June 1964).

3. Newell, A., editor, et al., Information Processing Language-V Manual, 2nd Edition, Prentice-Hall, Inc. (1964).

ACKNOWLEDGMENTS

We wish to thank the human opponents who played checkers with the program, and Mr. Leo Levitt, who furnished expert commentary on the game of checkers and on several of the games played by the program.

REFERENCES

1. Samuel, A. L., Some Studies in Machine Learning Using the Game of Checkers, IBM Journal of Research and Development 3, No. 3 (July 1959).

2. Hodges, Donald, IPL-VC, A Computer System Having the IPL-V Instruction Set, Argonne National Laboratory, ANL-6888 (June 1964).

3. Newell, A., editor, et al., Information Processing Language-V Manual, 2nd Edition, Prentice-Hall, Inc. (1964).